Unrestricted $LR(k)$ Grammars and its Parser, where $k=0,1$ (New Developments of Theory of Computation and Algorithms)

AUTHOR(S):

Shiina, Hiromitsu; Masuyama, Shigeru

数理解析研究所講究録 1205 巻 2001 年 154-159

Unrestricted $LR(k)$ Grammars and its Parser, where $k = 0, 1$

椎名広光 (Hiromitsu Shiina):
岡山理科大学 (Okayama University of Science)
増山繁 (Shigeru Masuyama):
豊橋技術科学大学 (Toyohashi University of Technology)

1 Introduction

Many parsers were developed for context-free languages. However, we often need to parse non context-free languages in natural language processing. Some parsers [1][2][4] were developed for classes larger than that of context-free languages. Vold'man[1] and Harris[2] developed parsers for the unrestricted grammar.

Along this line, we proposed the unrestricted $LR(k)$ grammar and the unrestricted $LR(k)$ parser [4]. The conventional $LR(k)$ grammar [3] has lookahead strings which consists of terminal symbols. On the other hand, in the unrestricted $LR(k)$ grammar, we proposed to regard lookahead strings as nonterminal symbols corresponding to roots of subtrees of deducible parse trees. Note that the unrestricted $LR(k)$ parser also works deterministically by using lookahead strings and parses the language $L(G)$ where it is in the class of recursive languages.

In this paper, we shall show how to transform an unrestricted $LR(k)$ grammar to an unrestricted $LR(1)$ grammar for any $k \geq 2$, which implies that there is an equivalent unrestricted $LR(1)$ grammar for any unrestricted $LR(k)$ grammar, $k \geq 2$, in that they generate the same language. A conventional LR grammar where generation capacity of $LR(k), k \geq 2$, grammar also coincides with $LR(1)$ grammar when k is finite[3]. However, it is interesting that this is also true even when $k = \infty$ when we consider the unrestricted LR grammar. Moreover, we clarify that a language generated by any recursive phrase structure language can be generated by some unrestricted $LR(k)$ grammar, where $k = 0, 1$.

2 Definitions

We define the unrestricted $LR(k)$ grammar, $G = (N, T, P, S)$, where N is a finite set of nonterminal symbols, T is a finite set of terminal symbols, P is a finite set of production rules and S is a start symbol. As preliminaries, we define the extended $LR(0)$ state transition diagram(LTD for short), the reachability state transition diagram (RTD for short) and lookahead strings where LTD, RTD and lookahead strings are used in the definition of the unrestricted $LR(k)$ grammar.

2.1 Definition of an LTD

An extended $LR(0)$ state transition diagram (LTD for short) is constructed from production rules of a phrase structure grammar $G = (N, T, P, S)$, which specifies a control part of a pushdown automaton. This LTD extends the $LR(0)$ state transition diagram to the phrase structure grammar. Basically, the creation algorithm is the same.

An LTD is constructed by a 6-tuple $M_L = (Q_L, \delta_s, \delta_r, s_0, s_{acc}, S')$. Here, $Q_L = \{s_0, s_1, \ldots, s_{n-1}, s_n\}$ is a set of states, δ_s is a set of shift functions satisfying $\delta_s(s, X) = s'$, $s, s' \in Q_L$, $X \in N \cup T$, δ_r is a set of reduce functions satisfying $\delta_r(s, X) = \{(\lambda \to \mu, s') | \ s' \in Q_L, \lambda \to \mu \in P\}$, s_0 is an initial state, s_{acc} is a final state and for convenience' sake let $s_{acc} = s_n$, and S' is a dummy symbol to construct an LTD in an initial step.

A state $s_i \in Q_L$, $i = 0, 1, \ldots, n$, is constructed from items called $LR(0)$ items. An $LR(0)$ item consists of production rules with a dot $(\cdot)$ like $[\lambda \to \cdot\mu_1\mu_2]$, $[\lambda \to \mu_1 \cdot \mu_2]$ and $[\lambda \to \mu_1\mu_2\cdot]$. A set of states and functions of LTD are constructed by the following five actions. Note that the length of string μ is shown as $|\mu|$, and an empty word is shown as ε.

[Action A: Initial action] As an initial action, we append a dummy rule $S' \to S$ to P, define an initial state s_0 and a final state s_{acc}.

$$s_0 := \{[S' \to \cdot S]\},$$
$$s_{acc} := \{[S' \to S\cdot]\} \text{ and}$$
$$\delta_s(s_0, S) := s_{acc}.$$

[Action B] If $[\lambda \to \cdot\mu_1 \cdot X\mu_2] \in s_i$ where $s_i \in Q_L$, $\mu_1, \mu_2 \in (N \cup T)^*$, then

- for each $X\xi \to \eta \in P$ where $\xi, \eta \in (N \cup T)^+$,
  $s_i := s_i \cup \{[X\xi \to \cdot\eta]\}$.

- for each $s_j \in Q_L$ such that $[\lambda \to \mu_1 X \cdot \mu_2] \notin s_j$,

create a new state s_k,
$Q_L := Q_L \cup \{s_k\}$ and
$s_k := s_k \cup \{[\lambda \to \mu_1 X \cdot \mu_2]\}$.

[Action C] If $[\lambda \to \mu_1 \cdot X\mu_2] \in s_i$, $[\lambda \to \mu_1 X \cdot \mu_2] \in s_j$ where $s_i, s_j \in Q_L$ and $\mu_1, \mu_2 \in (N \cup T)^*$, then

$$\delta_s(s_i, X) := s_j.$$

[Action D] If $[\lambda \to \mu \cdot X] \in s_i$, $[\lambda \to \mu X \cdot] \in s_j$ $[\lambda \to \cdot \mu X] \in s_k$ and $\delta_s(\delta_s(\ldots\delta_s(\delta_s(s_k, \mu_1), \mu_2)\ldots\mu_{|\mu|}), X) = s_i$ where s_i, s_j, $s_k \in Q_L$, $\mu = \mu_1\ \mu_2\ \cdots\ \mu_{|\mu|}$, $\mu_1, \mu_2, \ldots, \mu_{|\mu|} \in N \cup T$, then

$$\delta_r(s_i, X) := \delta_r(s_i, X) \cup \{(\lambda \to \mu X, s_k)\}.$$

[Action E] If $[\lambda \to \cdot\varepsilon] \in s_i$ where $s_i \in Q_L$, then

$$s_i := s_i \cup \{[\lambda \to \varepsilon \cdot]\} \text{ and}$$
$$\delta_r(s_i, \varepsilon) := \delta_r(s_i, \varepsilon) \cup \{(\lambda \to \varepsilon, s_i)\}.$$

Here, we use a grammar $G_1 = (\ \{S, A, B, C, D, E\}, \{S \to DE, DE \to ED, E \to ABB, E \to CB, A \to AA, A \to a, C \to CA, C \to a, B \to b, D \to d\ \}, \{a, b, d\}, S\ )$ as a working example. We can construct an LTD $M_L = (Q_L, \delta_s, \delta_r, s_0, s_{acc}, S')$ for the grammar G_1 where $Q_L = \{s_0, s_1, \ldots, s_{14}, s_{acc}\}$ and function δ_s and δ_r are shown in Tables 1 and 2.

In addition, contents of states are the following $LR(0)$ items,

$$s_0 = \left\{ \begin{array}{ll} [S' \to \cdot S], & [S \to \cdot DE], \\ [DE \to \cdot ED], & [E \to \cdot ABB], \\ [E \to \cdot CB], & [A \to \cdot AA], \\ [A \to \cdot a], & [C \to \cdot AC], \\ [C \to \cdot a] & [D \to \cdot d] \end{array} \right\},$$

$$s_1 = \{\ [A \to a \cdot],\ [C \to a \cdot]\ \},$$

$$s_2 = \left\{ \begin{array}{ll} [S \to D \cdot E], & [E \to \cdot ABB] \\ [E \to \cdot CB], & [A \to \cdot AA] \\ [A \to \cdot a], & [C \to \cdot AC] \\ [C \to \cdot a], & \end{array} \right\},$$

$\cdots\cdots\cdots,$

$$s_{acc} = \{\ [S' \to S \cdot]\ \}.$$

2.2 Definition of an RTD

In order to construct an RTD, we consider a push-down automaton which begins with s_i of an LTD by a nonterminal symbol X and stops at s_j of an LTD. Then we create a transition from s_i to s_j by X in an RTD. In short, we write $s_i \xrightarrow{X} s_j$. An RTD is constructed by this transition for all pairs of states of an LTD and nonterminal symbols, and is nondeterministic automaton.

Table 1: Function δ_s for G_1.

$\delta_s(s_0, S) = s_{acc}$	$\delta_s(s_0, a) = s_1$
$\delta_s(s_0, D) = s_2$	$\delta_s(s_0, E) = s_4$
$\delta_s(s_0, A) = s_7$	$\delta_s(s_0, C) = s_{11}$
$\delta_s(s_2, E) = s_3$	$\delta_s(s_2, A) = s_7$
$\delta_s(s_2, C) = s_{11}$	
$\delta_s(s_4, D) = s_5$	$\delta_s(s_4, d) = s_6$
$\delta_s(s_4, E) = s_4$	$\delta_s(s_4, A) = s_7$
$\delta_s(s_4, C) = s_{11}$	
$\delta_s(s_7, A) = s_8$	$\delta_s(s_7, B) = s_9$
$\delta_s(s_7, b) = s_{13}$	$\delta_s(s_7, a) = s_1$
$\delta_s(s_8, A) = s_8$	$\delta_s(s_8, a) = s_1$
$\delta_s(s_8, B) = s_{10}$	$\delta_s(s_8, C) = s_{12}$
$\delta_s(s_9, B) = s_{10}$	$\delta_s(s_9, b) = s_{13}$
$\delta_s(s_{11}, B) = s_{14}$	$\delta_s(s_{11}, b) = s_{13}$

An RTD M_R is a 4-tuple $(Q_R, \delta_R, s_0, s_{acc})$ where Q_R is a finite set of states of M_R, δ_R is a finite set of functions, s_0 is an initial state, s_{acc} is a final state. A set of states Q_R and a set of functions δ_R is defined by the following **[shift action]** and **[reduce action]**, respectively.

[shift action] If $(s_i, X) \vdash (s_j, \varepsilon)$ where $s_i, s_j \in Q_L$ and $X \in N$, then

$$Q_R := Q_R \cup \{s_i, s_j\},$$
$$\delta_R(s_i, X) := \delta_R(s_i, X) \cup \{s_j\}.$$

[reduce action] If $(s_i, X) \vdash (s_j, \alpha) \vdash\!\!\!\!+ (s_k, \varepsilon)$ where $s_i, s_j, s_k \in Q_L$, $(\alpha \to \beta X, s_j) \in \delta_r(s_i, X)$, $\alpha \to \beta X \in P$, $\alpha \in (N \cup T)^+$, $\beta = \beta_1\beta_2\ldots\beta_{|\beta|}$, $\beta_i \in N \cup T$ and $X \in N \cup \{\varepsilon\}$, then

$$Q_R := Q_R \cup \{s_i, s_k\},$$
$$\delta_R(s_i, X) := \delta_R(s_i, X) \cup \{s_k\}.$$

Among the above actions, we use the following three relations (s_i, α), $\vdash$ and $\vdash\!\!\!\!+$:

- (s_i, α) denotes that the current state of an LTD is s_i and a next input symbol $\alpha \in (N \cup T)^*$.
- $\vdash$ denotes a transition of a pair of a state of an LTD and next input symbols. If $\delta_s(s_i, X) = s_j$ and $(\alpha \to \beta X, s_j) \notin \delta_r(s_i, X)$, then $(s_i, X\alpha) \vdash (s_j, \alpha)$. On the other hand, if $(\alpha \to \beta X, s_j) \in \delta_r(s_i, X)$, then $(s_i, X\gamma) \vdash (s_j, \alpha\gamma)$.
- $\vdash\!\!\!\!+$ denotes a finite number of transitions of a pair consisting of a state of an LTD.

As an example of the RTD, we illustrate an RTD $M_R = (Q_R, \delta_R, s_0, s_{acc})$ of the grammar G_1.

First, we have $Q_R = \{s_0, s_2, s_4, s_7, s_8, s_9, s_{11}, s_{acc}\}$.

Table 2: Function δ_r for G_1.

$\delta_r(s_0, a) = \{(A \to a, s_0)\}$
$\delta_r(s_2, E) = \{(S \to DE, s_0)\}$
$\delta_r(s_2, a) = \{(A \to a, s_2)\}$
$\delta_r(s_4, E) = \{(DE \to ED, s_0), (DE \to ED, s_4)\}$
$\delta_r(s_4, a) = \{(A \to a, s_4), (C \to a, s_4)\}$
$\delta_r(s_4, d) = \{(D \to d, s_4)\}$
$\delta_r(s_7, A) = \{(A \to AA, s_0), (A \to AA, s_2),$
$(A \to AA, s_4)\}$
$\delta_r(s_7, a) = \{(A \to a, s_7), (C \to a, s_7)\}$
$\delta_r(s_7, b) = \{(B \to b, s_7)\}$
$\delta_r(s_7, C) = \{(C \to AC, s_0),$
$(C \to AC, s_2), (C \to AC, s_4)\}$
$\delta_r(s_8, A) = \{(A \to AA, s_0), (A \to AA, s_2),$
$(A \to AA, s_4), (A \to AA, s_7), (A \to AA, s_8)\}$
$\delta_r(s_8, a) = \{(A \to a, s_8), (C \to a, s_8)\}$
$\delta_r(s_8, C) = \{(C \to AC, s_7), (C \to AC, s_8)\}$
$\delta_r(s_9, B) = \{(E \to ABB, s_0), (E \to ABB, s_2),$
$(E \to ABB, s_4)\}$
$\delta_r(s_9, b) = \{(B \to b, s_9)\}$
$\delta_r(s_{11}, B) = \{(E \to CB, s_0), (E \to CB, s_2),$
$(E \to CB, s_4)\}$
$\delta_r(s_{11}, b) = \{(B \to b, s_{11})\}$

- For $s_0 \in Q_R$, we are able to consider the transitions $(s_0, S) \vdash (s_{acc}, \varepsilon)$, $(s_0, D) \vdash (s_2, \varepsilon)$, $(s_0, E) \vdash (s_4, \varepsilon)$, $(s_0, A) \vdash (s_7, \varepsilon)$ and $(s_0, C) \vdash (s_{11}, \varepsilon)$.

  We have five function values of δ_R from the above five transitions, $\delta_R(s_0, S) = \{s_{acc}\}$, $\delta_R(s_0, D) = \{s_2\}$, $\delta_R(s_0, E) = \{s_4\}$, $\delta_R(s_0, A) = \{s_7\}$, $\delta_R(s_0, C) = \{s_{11}\}$.

- For $s_7 \in Q_R$, we are able to consider transitions $(s_7, A) \vdash (s_8, \varepsilon)$, $(s_7, A) \vdash (s_0, A) \vdash (s_7, \varepsilon)$, $(s_7, A) \vdash (s_2, A) \vdash (s_7, \varepsilon)$, $(s_7, A) \vdash (s_4, A) \vdash (s_7, \varepsilon)$ and $(s_7, B) \vdash (s_9, \varepsilon)$.

  And, we have the following two function values of δ_R, $\delta_R(s_7, A) = \{s_4, s_7, s_8\}$, $\delta_R(s_7, B) = \{s_9\}$.

We illustrates a RTD in Fig. 1. Note that $\delta_R(s_i, X) \in s_j$ is shown as a path from s_i to s_j with label X ($s_i \xrightarrow{X} s_j$).

2.3 Definition of lookahead strings

A set of lookahead strings of length k is constructed as follows, using an RTD M_R.

$$LK_k([\alpha_1 \to \cdot\alpha_2], s_i) = \left\{ head_k(w\$^*) \,\middle|\, \begin{array}{l} s_i \xrightarrow[+]{\alpha_1} s_j, \\ s_j \xrightarrow[+]{w} s_{acc} \end{array} \right\}$$

The above formula shows that an unrestricted $LR(k)$ parser needs a nonterminal string w as a lookahead string to reduce "$\alpha_1 \to \alpha_2$", a nonterminal string w is necessary for a transition from a state s_j to a final state s_{acc} on an RTD.

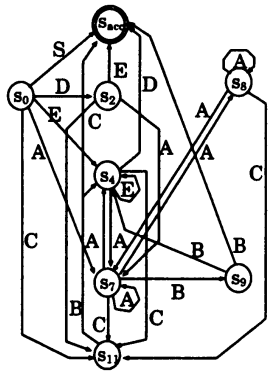

Figure 1: The RTD of G_1

Here $head_k(\lambda)$ consists of the prefix string of length k, and is defined by the following formula:

$$head_k(\lambda) = \begin{cases} \xi, & \text{if } \lambda = \xi\eta, |\xi| = k, |\eta| \geq 0, \\ \lambda, & \text{if } |\lambda| \leq k, \end{cases}$$

where $\lambda, \xi, \eta \in (N \cup T)^*$.

In addition, $s_i \xrightarrow[+]{\alpha} s_k$ means a transition from s_i to s_k by a nonterminal string α, it denotes $\delta_R(s_i, A_1) \ni s_{i_2}$, $\delta_R(s_{i_2}, A_2) \ni s_{i_3}$, ..., $\delta_R(s_{i_{|\alpha|}}, A_{|\alpha|}) \ni s_k$ where $\alpha = A_1 A_2 \ldots A_{|\alpha|}$, $A_1, A_2, \ldots, A_{|\alpha|} \in N$ and $s_i, s_{i_2}, \ldots, s_{i_{|\alpha|}}, s_k \in Q_R$.

For example, lookahead string LK_2 with length 2 is calculated as follows.

$$LK_2([DE \to \cdot ED], s_0) = \left\{ head_2(\varepsilon\$^*) \,\middle|\, \begin{array}{l} s_0 \xrightarrow[+]{DE} s_{acc}, \\ s_{acc} \xrightarrow[+]{\varepsilon} s_{acc} \end{array} \right\} = \{\$\$\ \}.$$

The transition from a start state s_0 with the input string DE in the above formula reaches a final state s_{acc}. Therefore, an unrestricted $LR(k)$ parser does not need an input string, that is, its lookahead string is sufficient with an empty word.

On the other hand, the transition from a start state s_0 with input string E reaches a state s_4 in the following formula. Therefore, w is necessary for the transition from the state s_4 to the final state s_{acc}.

$$LK_2([E \to \cdot CB], s_0) = \left\{ head_2(w\$^*) \,\middle|\, \begin{array}{l} s_0 \xrightarrow[+]{E} s_4, \\ s_4 \xrightarrow[+]{w} s_{acc} \end{array} \right\}$$
$$= \{\text{AA, AB, AC, CB, D\$, EA, EC, EE }\}.$$

For each $LR(0)$ item of s_0, if each length of lookahead strings is two, then we get lookahead strings LK_2's as follows.

$LK_2([S \to \cdot DE], s_0) = \{\$\$\}$
$LK_2([DE \to \cdot ED], s_0) = \{\$\$\}$
$LK_2([E \to \cdot ABB], s_0) = \{AA, AB, AC, CB, D\$, EA, EC, ED, EE\}$
$LK_2([E \to \cdot CB], s_0) = \{AA, AB, AC, CB, D\$, EA, EC, ED, EE\}$
$LK_2([A \to \cdot AA], s_0) = \{AA, AB, AC, BB, CB\}$
$LK_2([A \to \cdot a], s_0) = \{AA, AB, AC, BB, CB\}$
$LK_2([C \to \cdot AC], s_0) = \{BA, BC, BD, BE\}$
$LK_2([C \to \cdot a], s_0) = \{BA, BC, BD, BE\}$

On the other hand, for the case of each length of the lookahead strings is one, we have the following equations.

$LK_1([A \to \cdot a], s_0) = \{A, B, C\}$ and $LK_1([C \to \cdot a], s_0) = \{B\}$.

2.4 Definitions for an unrestricted $LR(k)$ grammar

First, for a phrase structure grammar $G = (N, T, P, S)$, we construct an LTD. And if all states of an LTD does not have the item like $[\alpha_1 \to \alpha_2 \cdot]$ or $[\beta_1 \to \beta_2 \cdot]$ where dot$(\cdot)$ is the end of a production rule in the same state of an LTD, then the grammar G is defined as an unrestricted $LR(0)$ grammar. On the other hand, if a state of an LTD has an $LR(0)$ item, the grammar G is defined as an unrestricted $LR(k)$ grammar, $k \geq 1$, which satisfies the following two conditions:

- **(Condition 1)** Let an LTD M_L of an unrestricted $LR(0)$ grammar be $(Q_R, \delta_R, s_0, s_{acc}, S')$. Each production in P has either of the following three forms:
  (a) $A \to a$, $A \in N$, $a \in T$.
  (b) $A \to \varepsilon$, $A \in N$, where ε is the empty word.
  (c) $\lambda \to \mu$, $\lambda, \mu \in N^+$.
- **(Condition 2)** Each production $\alpha_i \to \beta \in P$, $\alpha_i \in N^+$, exists in the same state $s_j \in Q_L$, and LK_k satisfies

$$\bigcap_{\alpha_i : \alpha_i \to \beta \in P} LK_k([\alpha_i \to \cdot\beta], s_j) = \emptyset.$$

Condition 1 is not essential as transformation , because any phrase structure grammar can be transformed to a grammar satisfying **condition 1** which generates the same language.

For the grammar G_1 defined in section 2.1, $LK_1([A \to \cdot a], s_0) \cap LK_1([C \to \cdot a], s_0) = \{B\}$. Therefore, G_1 is not an unrestricted $LR(1)$ grammar. On the other hand, no string is shared among lookahead strings of length 2 (see Table 3). Therefore, G_1 is not an unrestricted $LR(1)$ grammar but an unrestricted $LR(2)$ grammar.

Table 3: The set of the lookahead strings LK_2 for G_1.

$LK_2([S \to \cdot DE], s_0) = \{\$\$\}$
$LK_2([DE \to \cdot ED], s_0) = \{\$\$\}$
$LK_2([E \to \cdot ABB], s_0) = \{AA, AB, AC, CB, D\$, EA, EC, ED, EE\}$
$LK_2([E \to \cdot CB], s_0) = \{AA, AB, AC, CB, D\$, EA, EC, ED, EE\}$
$LK_2([A \to \cdot AA], s_0) = \{AA, AB, AC, BB, CB\}$
$LK_2([A \to \cdot a], s_0) = \{AA, AB, AC, BB, CB\}$
$LK_2([C \to \cdot AC], s_0) = \{BA, BC, BD, BE\}$
$LK_2([C \to \cdot a], s_0) = \{BA, BC, BD, BE\}$
$LK_2([E \to \cdot ABB], s_2) = \{\$\$\}$
$LK_2([E \to \cdot CB], s_2) = \{\$\$\}$
$LK_2([A \to \cdot AA], s_2) = \{AA, AB, AC, AD, AE, BB, CB\}$
$LK_2([A \to \cdot a], s_2) = \{AA, AB, AC, AD, AE, BB, CB\}$
$LK_2([C \to \cdot AC], s_2) = \{BA, BC, BD, BE\}$
$LK_2([C \to \cdot a], s_2) = \{BA, BC, BD, BE\}$
$LK_2([DE \to \cdot ED], s_4) = \{\$\$\}$
$LK_2([D \to \cdot d], s_4) = \{\$\$\}$
$LK_2([E \to \cdot ABB], s_4) = \{AA, AB, AC, CB, D\$, EA, EC, ED, EE\}$
$LK_2([E \to \cdot AC], s_4) = \{AA, AB, AC, CB, D\$, EA, EC, ED, EE\}$
$LK_2([A \to \cdot AA], s_4) = \{AA, AB, AC, AD, AE, BB, CB\}$
$LK_2([A \to \cdot a], s_4) = \{AA, AB, AC, AD, AE, BB, CB\}$
$LK_2([C \to \cdot AC], s_4) = \{BA, BC, BD, BE\}$
$LK_2([C \to \cdot a], s_4) = \{BA, BC, BD, BE\}$
$LK_2([A \to \cdot AA], s_7) = \{AA, AB, AC, AD, AE, BB, CB\}$
$LK_2([A \to \cdot a], s_7) = \{AA, AB, AC, AD, AE, BB, CB\}$
$LK_2([B \to \cdot b], s_7) = \{BA, BC, BD, BE\}$
$LK_2([C \to \cdot AC], s_7) = \{BA, BC, BD, BE\}$
$LK_2([C \to \cdot a], s_7) = \{BA, BC, BD, BE\}$
$LK_2([A \to \cdot AA], s_8) = \{AA, AB, AC, AD, AE, BB, CB\}$
$LK_2([C \to \cdot AC], s_8) = \{BA, BC, BD, BE\}$
$LK_2([C \to \cdot a], s_8) = \{BA, BC, BD, BE\}$
$LK_2([B \to \cdot b], s_9) = \{AA, AB, AC, CB, D\$, EA, EC, ED, EE\}$
$LK_2([B \to \cdot b], s_{11}) = \{BA, BC, BD, BE, \$\$\}$

3 The transformation from an unrestricted $LR(k)$ grammar, $k > 1$, to an unrestricted $LR(1)$ grammar

The transformation [3] of a conventional $LR(k)$ grammar replaces a pair of a nonterminal symbol corresponding to a root of a subtree of deducible parse trees and a lookahead string of its nonterminal symbol with a nonterminal symbol. On the other hand, the transformation in this paper replaces a head of lookahead strings with a nonterminal symbol, and its replacement is repeated until the length of lookahead strings becomes 1. In this section, we illustrate how to transform an unrestricted $LR(k)$

grammar, $k > 1$, to an unrestricted $LR(1)$ grammar.

For an unrestricted $LR(k)$ grammar G, $k > l \geq 1$, a set of lookahead strings satisfies

$$LK_k([\lambda_1 \rightarrow \cdot\mu], s_i) \cap LK_k([\lambda_2 \rightarrow \cdot\mu], s_i) = \emptyset \text{ and}$$
$$LK_l([\lambda_1 \rightarrow \cdot\mu], s_i) \cap LK_l([\lambda_2 \rightarrow \cdot\mu], s_i) \ni \alpha$$

where $\alpha = A\alpha', \alpha, \alpha' \in (N \cup T)^+$ and $A \in N$. Clearly, we have $LK_1([\lambda_1 \rightarrow \cdot\mu], s_i) \cap LK_1([\lambda_2 \rightarrow \cdot\mu], s_i) \ni A$.

To reduce from a lookahead string α to a nonterminal symbol, we prepare a new nonterminal symbol $B \notin N$ and replace production rules so that $LK_1([\lambda_1 \rightarrow \cdot\mu], s_i) \ni B$ and $LK_1([\lambda_2 \rightarrow \cdot\mu], s_i) \ni A$. The following transformation describes precise conditions and replacement of production rules.

[Reduction of multiple symbols]
If $LK_1([\mu_1 \rightarrow \cdot\beta], s_i) \cap LK_1([\mu_2 \rightarrow \cdot\beta], s_i) \ni A$ where $\mu_1 \rightarrow \beta$, $\mu_2 \rightarrow \beta \in P$, $s_i \in Q_R$, $s_j \in \{s_k | s_i \stackrel{\mu_1}{\rightarrow} s_k, s_k \in Q_R\}$, $\{[\alpha \rightarrow \alpha_1 \cdot A\gamma_i]\} \in s_j$ and $\gamma_i = \alpha_{i+1}\gamma'$, then

$$(1)P := P - \{\alpha \rightarrow \alpha_1 A\gamma_i\}.$$
$$(2)P := P \cup \{\alpha \rightarrow \alpha_1 B\gamma_i, B\alpha_{i+1} \rightarrow \gamma_{i+1}, B\alpha_{i+2} \rightarrow \gamma_{i+2}, \ldots, B\alpha_{n_2} \rightarrow \gamma_{n_2}\}.$$
$$(3)N := N \cup \{B\}.$$

The above transformation is repeated until the grammar G becomes an unrestricted $LR(1)$ grammar, i.e., until the following condition is satisfied:
For each $\beta \in N^*$, $s_i \in Q_R$,

$$\bigcap_{\alpha:\alpha\rightarrow\beta\in P} LK_1([\alpha \rightarrow \cdot\beta], s_i) = \emptyset.$$

Note that the above transformation does not depend on the length k of a lookahead string and works even if $k = \infty$.

4 Properties of a transformation

In the above section, we showed the transformation of an unrestricted $LR(k)$ grammar. In this section, we shall show some properties of the transformation.

Theorem 1: The transformation of the grammar introduced in section 3 does not change the language L generated by the grammar if L is in the class of recursive languages.

Proof. Let G' be the resulting grammar obtained from a grammar G, by an application of **[Reduction of multiple symbols]** in section 3, where $\alpha \rightarrow \alpha_1 A\gamma_i$ is deleted and $\alpha \rightarrow \alpha_1 B\gamma_i$ and $B\alpha_{i+1} \rightarrow \gamma_{i+1}$ are appended for production rules $\alpha \rightarrow \alpha_1 A\gamma_i$, $A\alpha_{i+1} \rightarrow \gamma_{i+1}$ of G, where $\gamma_i = \alpha_{i+1}\gamma'$. Then we shall show that $L(G) = L(G')$.

We consider the following derivation from a string $w_1\alpha w_2$, by production rules $\alpha \rightarrow \alpha_1 A\gamma_i$ and $A\alpha_{i+1} \rightarrow \gamma_{i+1}$ of G.

$$w_1\alpha w_2 \Rightarrow w_1\alpha_1 A\gamma_i w_2 = w_1\alpha_1 A\alpha_{i+1}\gamma' w_2 \Rightarrow w_1\alpha_1\gamma_{i+1}\gamma' w_2$$

We also consider the derivation shown as follows by production rules $\alpha \rightarrow \alpha_1 B\gamma_i$ and $B\alpha_{i+1} \rightarrow \gamma_{i+1}$ of G'.

$$w_1\alpha w_2 \Rightarrow w_1\alpha_1 B\gamma_i w_2 = w_1\alpha_1 B\alpha_{i+1}\gamma' w_2 \Rightarrow w_1\alpha_1\gamma_{i+1}\gamma' w_2$$

Hence, the same strings are derived from $w_1\alpha_i w_2$ by G and G', respectively. Thus we conclude that $L(G) = L(G')$ if $L(G)$ is in the class of recursive languages. By repeating the above discussion m times where m is the number of applications of **[Reduction of multiple symbols]** in section 4 to obtain an unrestricted $LR(1)$ grammar, we have shown that this theorem holds. □

Theorem 2: Any phrase structure grammar G where $L(G)$ is in the class of recursive languages can be transformed to either an unrestricted $LR(0)$ grammar or an unrestricted $LR(1)$ grammar.

Proof. By the definition of the unrestricted $LR(k)$ grammar, it is easy to see that we can construct an unrestricted $LR(k)$ grammar for any phrase structure grammar which generates the same language if we allow k to be ∞. The transformation from an unrestricted $LR(k)$, $k > 1$, grammar to an unrestricted $LR(1)$ grammar introduced in section 4 works even if $k = \infty$, as the transformation procedure does not depend on k. Thus consider the unrestricted $LR(k)$ grammar G which generates the same language as the given phrase structure grammar. If $k > 1$, then, by Theorem 1, there is an unrestricted $LR(1)$ grammar which generates the same language as the given phrase structure language when it is in the class of recursive languages. The case of $k = 0$ is trivial. Thus we have proved the theorem. □

5 An example of the transformation and parsing

The lookahead strings LK_1's of the grammar G_1 consist of

$$LK_1([A \rightarrow \cdot a], s_0) = \{A, B, C\} \text{ and}$$
$$LK_1([C \rightarrow \cdot a], s_0) = \{B\}.$$

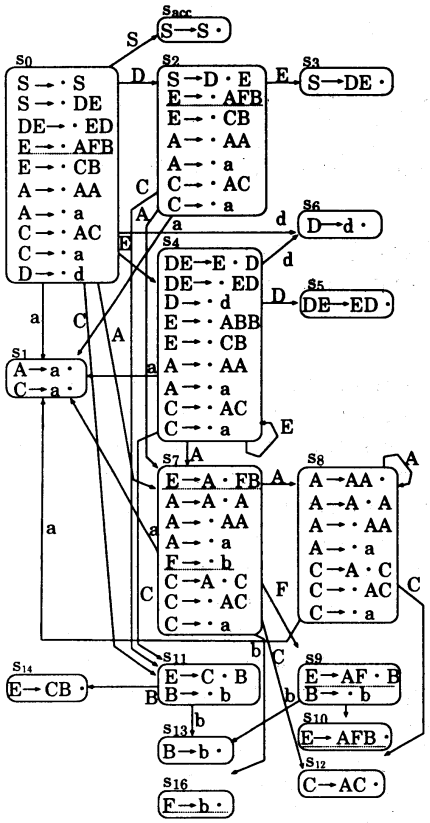

Figure 2: The LTD of G_2

From the above lookahead strings, we have the following equation.

$$LK_1([A \to \cdot a], s_0) \cap LK_1([C \to \cdot a], s_0) = \{B\}.$$

Hence, to reduce each length of the lookahead strings, we prepare a new nonterminal symbol $F \notin N_1$ and replace B with F such that

$$LK_1([A \to \cdot a], s_0) = \{A, F, C\},$$
$$LK_1([C \to \cdot a], s_0) = \{B\}.$$

From the RTD of G_1 (see Fig. 1), we have $\delta_R(A, s_0) = s_7$ and $\delta_R(C, s_0) = s_{11}$. Therefore, there is B followed by a dot $(\cdot)$ in an $LR(0)$ item of s_7. In fact, there is the $LR(0)$ item $[E \to A \cdot BB]$ in s_7. Consequently, we replace a production rule $E \to ABB$ with production rules $E \to AFB$ and $F \to b$. Here a new nonterminal symbol F is followed by a dot $(\cdot)$ in an $LR(0)$ item.

- $P := P - \{E \to ABB\}$
- $P := P \cup \{E \to AFB, F \to b\}$
- $N := N \cup \{F\}$

By the above transformation, we have the following grammar G_2 = ($\{S, A, B, C, D, E, F\}$, $\{a, b, d\}$, $\{S \to DE, DE \to ED, E \to CB, A \to AA, A \to a, C \to CA, C \to a, B \to b, E \to AFB, F \to b, D \to d\}$, S). In addition, we define an example of the LTD in Fig. 2 and the RTD in Fig. 3. Clearly, $LK_1([A \to \cdot a], s_0) \cap LK_1([C \to \cdot a], s_0) = \emptyset$, in short, the grammar G_2 satisfies

$$\bigcap_{\alpha:\alpha\to\beta\in P} LK_1([\alpha \to \cdot\beta], s_i) = \emptyset.$$

The resulting grammar G_2 by the transformation is an unrestricted $LR(1)$ grammar.

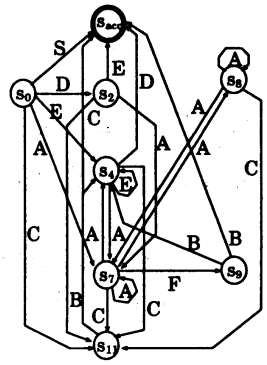

Figure 3: The RTD of G_2

6 Concluding Remarks

In this paper, we showed how to transform an unrestricted $LR(k)$ grammar to an unrestricted $LR(1)$ grammar without changing the generated language which is in th class of recursive languages. Although a conventional $LR(k)$, $k \geq 1$, grammar can be transformed to $LR(1)$ only when k is a finite integer. Even if $k = \infty$ as was shown in Theorem 2, we can trasform an unrestricted $LR(k)$, $k \geq 1$, grammar to an unrestricted $LR(1)$ grammar.

References

[1] G.Sh. Vold'man, *A parsing algorithm for context-sensitive grammars*, Program. Comput. Software, 7 (1981) 302-307.

[2] L.A. Harris, *SLR(1) and LALR(1) parsing for unrestricted grammar*, Acta Inform., 24 (1987) 191-209.

[3] S. Sippu and E.Soisalon-soininen, *Parsing Theory Vol. II:LR(k) and LL(k) Parsing*, Springer-Verlag, (1990).

[4] H. Shiina and S. Masuyama, *Proposal of the unrestricted LR(k) grammar and its parser*, Mathematica Japonica, 46 (1) (1997) 129-142.